Proceedings of the V Brazilian Conference on Neural Networks -V Congresso Brasileiro de Redes Neurais
pp. 541-546, April 2-5, 2001- Rio de Janeiro-RJ-Brazil

The Levenberg-Marquardt Method Applied in Nonlinear Dynamic Systems

Silva, A. P. C., Khater, E.
DEMEC/FUNREI
E-mails: lasid@funrei.br, khater@funrei.br

Abstract

This work consist in the identification of nonlinear dynamic systems using two functions with the backpropagation algorithm. Four optimization methods were implemented for a convergence comparation: Gradient, Momentum, Conjugate Gradient and Levenberg-Marquardt. The main objective was show, by the results, that the Levenberg-Marquardt method presented a better performance in the identification of the researched systems. Moreover, this work investigated the influence of the inputs in the network training performance.

1. Introduction

The modern era of neural networks began with the pioneering work of McCulloch and Pitt's (1943). During the classical period of the perceptron in the 1960s, it seemed as if neural networks could do anything. But then came the book by Minsky and Papert (1969), who used mathematics to demonstrate that there are fundamentals limits on what single-layer perceptron can compute.

It was need to wait until the 1980s for the solution of these basic problems to emerge. In 1982 Hopfield used the idea of an energy function to formulate a new way of understanding the computation performed by recurrent networks with symetric synaptic connections. This particular class of neural networks with feedback attracted a great deal of attention in the 1980s , and in the course of time it has come to be known as Hopfield networks.

In 1986 the development of backpropagation algorithm was reported by Rumelhart, Hinton, and Williams. This algorithm has emerged as the most popular learning algorithm for the training of multilayer perceptrons [1].

Since the backpropagation learning algorithm was first popularized, there has been considerable research on methods to accelerate the convergence of the algorithm. This research falls roughly into two categories. The first category involves the development of ad hoc techniques. These techniques include such areas as varying the learning rate, using momentum and rescaling variables. Another category of research has focused on standard numerical optimization techniques [2].

System identification has an extensive literature. The first detailed study of this subject using neural networks appeared in Narendra and Parthasarathy (1990). System identification is the experimental approach to the modeling of a process or a plant of unknown parameters [1].

For linear systems, if the order is known, the structure of the model can be chosen so that one is left with the task of parameter estimation. This does not, however, apply to nonlinear system identification, where the structure of the model has to be justified. Since the true system is not known, it must be assumed that it belongs to a specified set and that the parameterized model chosen can theoretically realize the input-output behavior of any system belonging to that class. Presented in this fashion, identification reduces to a parameter estimation problem [3].

Some of the advantages of using artificial neural networks as the model for system identification are: (a) ability to approximate arbitrary nonlinear funtionals to any degree of accuracy; (b) they are adaptive, thus they can take data and learn from it, often capturing subtle relationships; (c) they can handle corrupt or incomplete data, thus providing a measure of fault tolerance; and (d) they are highly parallel, which allows numerous independent operations to be executed simultaneously [4].

The comparation made in this work among the performance of the optimization methods in the identification of the two nonlinear functions aim at to define, by the results, wich of them present a fastest and more accurate convergence. So that, in futures works with nonlinear functions it will be possible apply directly the more efficient method here defined.

2. Backpropagation network

The backpropagation algorithm derives its name from the fact that the partial derivatives of the cost function (performance measure) with respect to the free parameters (synaptic weights and biases) of the network are determined by backpropagating the error signals (computed by the output neurons) through the network, layer by layer [1].

The main advantage of the backpropagation method is that the theaching performance is drastically improved by the introduction of the hidden layer. On the other hand, the main disadvantages of the backpropagation error lie in (a) proper selection of suitable system parameters which are necessary to reduce the time of learning and (b) the existence of a large number of local minima on the solution and the difficulty in determining the global optimum within it [5].

The backpropagation algorithm use as performance index the mean square error and is provided with a set of examples of proper network behavior [6]:

$$\{p_1, t_1\}, \{p_2, t_2\}, ..., \{p_q, t_q\} \qquad (1)$$

p_q – input to the network,

t_q – corresponding target output.

The algorithm should adjust the network parameters in order to minimize the mean square error. That way, the first step is to propagate the input forward through the network. The equations that describe this operation are:

$$a^0 = p, \qquad (2)$$

$$a^{m+1} = f^{m+1}\left(W^{m+1}a^m + b^{m+1}\right) \qquad (3)$$

for m=0, 2, ..., M-1,

$$a = a^M \qquad (4)$$

where
p – input vector,
a^{m+1} – output vector of the layer (m+1),
f^{m+1} – transfer functions of the layer (m+1),
W^{m+1} – weight matrix of the layer (m+1),
b^{m+1} – bias vector of the layer (m+1).

The next step is to propagate the sensitivities backward through the network:

$$s^M = -2\dot{F}^M\left(n^M\right)(t-a), \qquad (5)$$

$$s^m = \dot{F}^m\left(n^m\right)\left(W^{m+1}\right)^T\left(s^{m+1}\right) \qquad (6)$$

for M-1, ..., 2, 1.

where
s^M – sensitivities vector of the last layer. It is the starting point for the recurrence relationship.
s^m – sensitivities vector of a hidden layer.
$\dot{F}$ - first derivatives of the transfer function.

Finally, the weights and bias are updated using optimization algorithms.

3. Optimization methods

This work was done comparing the identification using four different methods. They were**:**

3.1. Gradient:

Based on the first-order Taylor series expansion having low convergence rate mainly in developed fases of the adjuste process, where the error is small [7].

This algorithm adjusts the wheights in the steepest descent direction (negative of the gradient). This is the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. The update is done using the following the equations:

$$W^m(k+1) = W^m(k) - \alpha s^m\left(a^{m-1}\right)^T, \qquad (7)$$

$$b^m(k+1) = b^m(k) - \alpha s^m. \qquad (8)$$

α - learning rate

3.2. Momentum:

This method allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Acting like a low pass filter, momentum allows the network to ignore small features in the error surface. Without momentum a network may get stuck in a shallow local minimum. With momentum a network can slide through such a minimum.

This algorithm is obtained increasing a momentum coefficient in the gradient equation in order to reduce the output signal amplitude [6]. The parameters update equations are:

$$\Delta W^m(k+1) = \gamma\,\Delta W^m(k-1) - (1-\gamma)\alpha\, s^m\left(a^{m-1}\right)^T, \qquad (9)$$

$$\Delta b^m(k) = \gamma\,\Delta b^m(k-1) - (1-\gamma)\alpha\, s^m. \qquad (10)$$

γ - mometum coefficient

3.3. Conjugate-Gradient:

Based on the second-order Taylor series expansion. It doesn’t require the calculation of the second derivatives. While in gradient algorithm the search directions at consecutive iterations are orthogonal, in the conjugate-gradient is done a sequence of exact linear searches along conjugate directions [6]. Its update equations are given below:

$$x_{k+1} = x_k + \alpha_k\, p_k, \qquad (11)$$

and

$$p_k = -g_k + \beta_k p_{k-1} \qquad (12)$$

where

x_k - vector that combine the weight matrix and the bias at the kth iteration.
p_k - search direction.
g_k - gradient avaluated at x_k.
β_k - scalar that can be chosen by different methods.

There are some variations of conjugate gradient algoritms. The various versions of the conjugate gradient are distinguished by the manner in which the constant β_k is computed.

In this work is used the Fletcher-Reeves update, where β_k is computed by the following form:

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \qquad (13)$$

3.4. Levenberg-Marquardt:

This algorithm is obtained by a modification in the Gauss-Newton method.

The weights update in the Gauss-Newton method is given by:

$$\Delta x_k = -\left[J^T(x_k)J(x_k)\right]^{-1} J^T(x_k)\nu(x_k) \qquad (14)$$

The Levenberg-Marquardt modification above the Gauss-Newton method consist in the increment of a parameter μ_k:

$$\Delta x_k = -\left[J^T(x_k)J(x_k) + \mu I\right]^{-1} J^T(x_k)\nu(x_k) \qquad (15)$$

where
$J(x_k)$ - jacobian matrix,
$\nu(x_k)$ - error vector.

When μ_k is increased it approaches the gradient algorithm with small learning rate, when μ_k is decreased to zero the algorithm becomes Gauss-Newton [2].

The algorithm begins with μ_k set to some small value. If a step does not yield a smaller value for the performance index, the step is repeated with μ_k multiplied by some factor $\vartheta > 1$. Eventually the performance index should decrease, since we would be taking a small step in direction of gradient. If a step does produce a smaller value for the performance index, then μ_k is divided by ϑ for the next step, so that the algorithm will approach Gauss-Newton, wich should provide faster convergence [6].

4. Results

In this work the dynamic systems are characterized by two nonlinear functions.

4.1. First function

The first function used in the identification is showed below:

$$y_p(k+1) = 0.3y_p(k) + 0.6y_p(k-1) + f[u(k)] \qquad (16)$$

$$f(u) = 0.6\sin(\pi \cdot u) + 0.3\sin(3 \cdot \pi \cdot u) + 0.1\sin(5 \cdot \pi \cdot u) \qquad (17)$$

$$u(k) = \sin(2 \cdot \pi \cdot k / 250) \qquad (18)$$

k = 1:700.

As a input was used the sinusoid:

$$u(k) = \sin(2 \cdot \pi \cdot k / 250) \qquad (19)$$

k = 1:700

As a stop criterion was used na error of 10^{-5} or 250000 of iterations .

The network architecture was fixed in:
- 1 input layer with 20 neurons.
- 1 hidden layer with 10 neurons.
- 1 output layer with 1 neuron.

The graphics are showing the identification obtained by the four differents methods.

- Gradient method

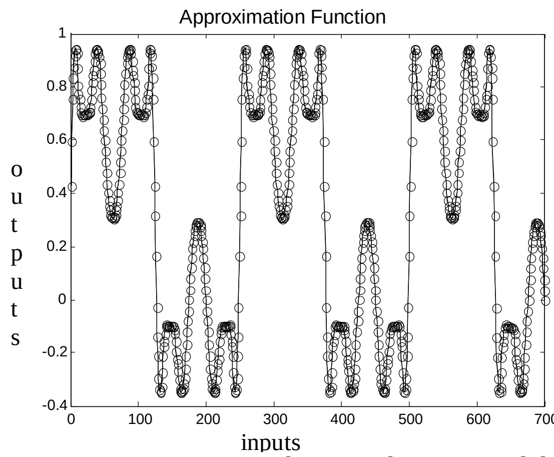

Figure 1 – Comparation between the target and the neural network output.

Legend: ____ - target -o-o - network output

- Momentum

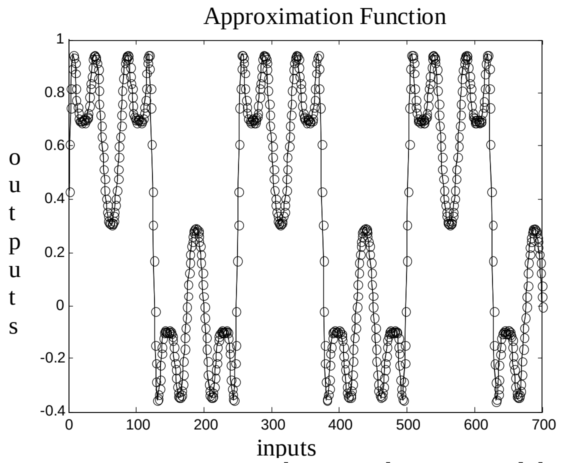

Figure 2 – Comparation between the target and the neural network output.

Legend: ____ - target -o-o - network output

- Conjugate Gradient

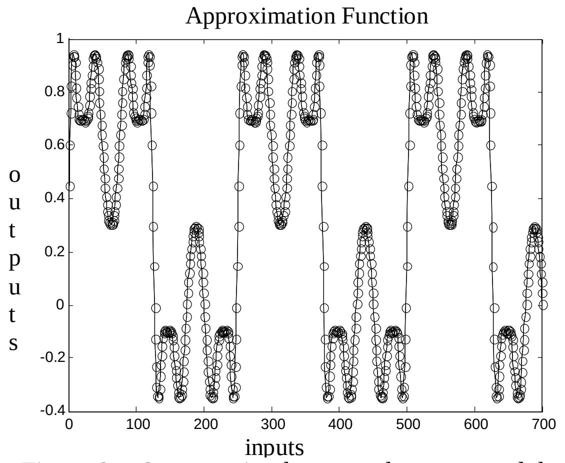

Figure 3 – Comparation between the target and the neural network output.

Legend: ____ - target -o-o - network output

- Levenberg-Marquardt

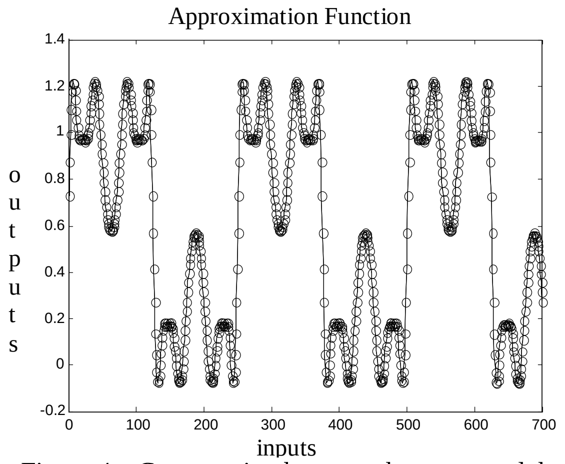

Figure 4 – Comparation between the target and the neural network output.

Legend: ____ - target -o-o - network output

The table 1 presents the datas obtained in the identification of this function allowing a clearest comparation of the methods performance.

Table 1: Main results

First Function			
Method	Error	Time	Iteration
Grad.	6.78970e-5	26hs	250000
Mom.	4.52908e-5	20hs	250000
Conj.	0.99915e-5	59m	4939
L. M.	0.96726e-5	2m	34

The figures 1 to 4 show the graphics of the identification provided by the Gradient, Momentum, Conjugate Gradient and Levenberg-Marquardt methods, respectively. Analysing these figures it's possible to notice that the curves representing the target and the network output are superposed indicating that was obtained na approximation in agreement with the expected. Howerver, according table 1 datas it is verified that the Gradient and Momentum methods presented similar results and stoped the training by reaching the number of iterations defined in the stop criterion. The Conjugate Gradient and Levenberg-Marquardt were fastest in this function identification and both reached the specified error as the stop criterion. The great gain in the Levenberg-Marquardt method was in the processing time, wich was significantly smaller than the others three methods.

4.2. Second function

The second function used in the identification is showed below:

$$y_p(k+1) = f\left[y_p(k), y_p(k-1)\right] + u(k) \qquad (20)$$

where

$$f\left[y_p(k), y_p(k-1)\right] = \frac{y_p(k)\cdot y_p(k-1)\cdot\left[y_p(k)+2.5\right]}{1+y_p^{\,2}(k)+y_p^{\,2}(k-1)} \qquad (21)$$

and

$$u(k) = \sin\left(2\cdot\pi\cdot k/25\right) \qquad (22)$$

k=1:102

As inputs were used two vectors 1x102 with aleatorys elements.

As a stop criterion was used na error of 10^{-5} or 1000000 of iterations.

The network architecture was fixed in:

- 1 input layer with 20 neurons.
- 1 hidden layer with 10 neurons.
- 1 output layer with 1 neuron.

The graphics are showing the identification obtained by the four differents methods.

- Gradient method:

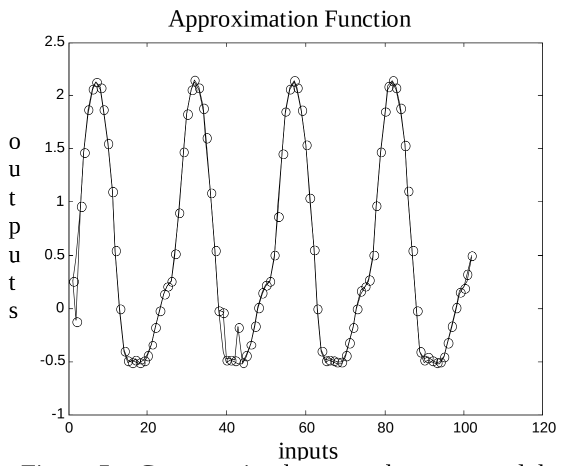

Figure 5 – Comparation between the target and the neural network output.

Legend: ____ - target -o-o - network output

- Momentum:

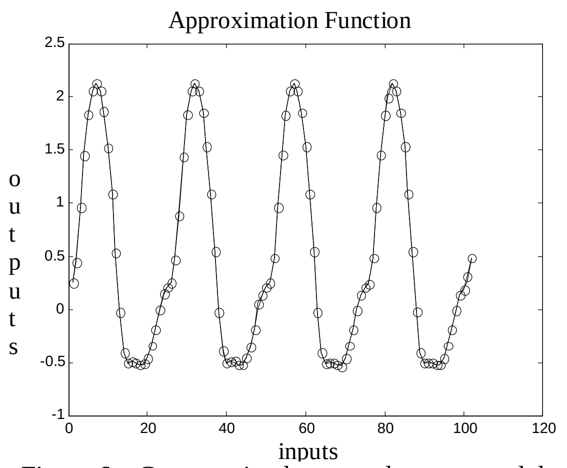

Figure 6 – Comparation between the target and the neural network output.

Legend: ____ - target -o –o – network output

- Conjugate-Gradient:

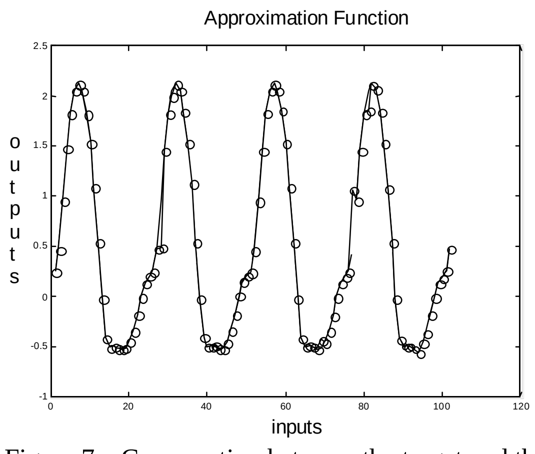

Figure 7 – Comparation between the target and the neural network output.

Legend: ____ - target -o-o - network output

- Levenberg-Marquardt:

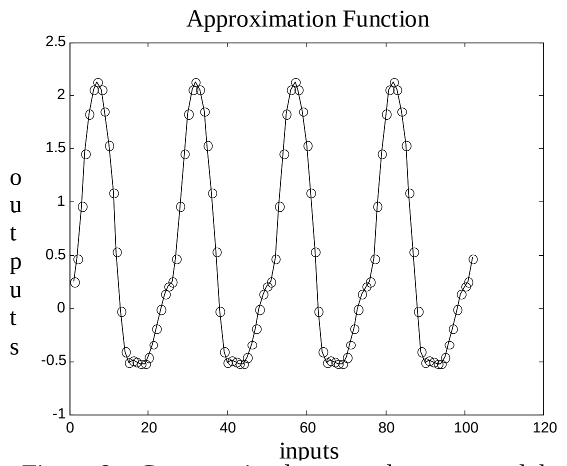

Figure 8 – Comparation between the target and the neural network output.

Legend: ____ - target -o- o – network output

The table 2 present the main datas obtained in these identifications allowing a better comparation among the methods.

Table 2: Main results

Second Function			
Method	Error	Time	Iteration
Grad.	650.38e-5	22hs	1000000
Mom.	18.59e-5	31hs	1000000
Conj.	608.22e-5	44hs	1000000
L. M.	0.48e-5	12m	311

The figure 5 show the identification provided by the Gradient method. Analysing this figure it's possible to notice that the curves representing the network output and the target aren't superposed in all the points. The table 2 show that this method stoped by reaching the number of iterations defined in the stop criterion and it didn't reach the error specified.

The figure 6 show the identification provided by the Momentum method. Observing this figure it is possible to see that the curves are superposed. However the table 2 show that this method didn't reach the specified error and stop the training by reaching the number of iterations defined in the stop criterion.

The figure 7 present the identification provided by the Conjugate Gradient method. Analysing carefully this graphic it's possible to notice that there is a little deviation in the curves superpose. According datas in table 2 this method stoped by reaching the number of iterations defined in the stop criterion and didn't reach the error specified.

The figure 8 present the identification provied by the Levenber-Marquardt method. In this graphic the curves are superposed. This method reached the specified error and, moreover, presented a computational cost significantly reduced in relation to the other three methods.

5. Conclusions

It was verified that the Levenber-Marquardt method was the most suitable method for the identification of this two studied functions. It represented a great redution in the computational cost in relation to the others three tested methods, reducing the implementation time and the number of iterations processed.

It was verified that the use of aleatories inputs increase the difficulty in the methods convergence, because in the second funtion training was need a high number of iterations and, nevertheless, the specified error wasn't reached in the Gradient, Momentum and Conjugate Gradiente methods. Only the Levenber-Marquardt method reached the specified error in a reduced time and number of iterations.

With the objective of analyse the inputs influence in the network training the two functions were trained with yours inputs changed, that is to say, the first function inputs were applied in the second function and vice-versa. The others parameters were maintained constants. With the modified inputs the curves of the real and target output didn't superpose and the error was close 10^{-1} in the two functions showing that the variation of a single parameter can influence in the all network identification capability.

The results obtained in this work are na incentive for the continuity in the performance analyse of the Levenberg-Marquardt method in others identification models, so that generalizations could be done in the sense of define wich parameters are better adaptables for certain functions classes.

References

[1] Haykin, S. Neural Networks: A Comprehensive Foundation. Prentice-Hall, 1999, 696p.

[2] Martin, T., Hagan, Mohammad, B. Menhaj. Training Feedforward Networks with the Marquardt Algorithm. IEEE Trans. on Neural Networks, v. 5, n. 2, p. 989-993. November 1994.

[3] Levin, U. Asriel, Narendra, S. Kumpati. Control of Nonlinear Dynamical Systems Using Neural Networks – Part II: Observability, Identification, and Control. IEEE Trans. on Neural Networks, v.7, n.1, p. 30-42. January 1996.

[4] Lou, Kang-Ning, Perez, Ronald A. A New System Identification Technique Using Kalman Filtering and Multilayer Neural Networks. Artificial Intelligence Engeering , 1996. P. 1-8.

[5] Paul, Chinmoy, Kayaba, Naoki, Morishita, Shin, Hagiwara, Ichiro. Dynamic System Identification by Neural Network (A New Fast Learning Method Based on Error Back Propagation). JSME International Journal, Series C, v. 38, n. 4, p. 686-692. 1995.

[6] Hagan, T. M., Demuth B. H., Beale, M. Neural Network Design. PWS Publishing Company, 1996.

[7] Khater, Evaldo. Controle de vibração torcional em sistemas rotativos usando redes neurais multicamadas. Campinas: Universidade Estadual de Campinas (UNICAMP), 1998. 126p. Tese de Doutorado.